The Bazaar World of Gaming

Members:

Arjun Garg (garg14)

Ben Levy (bllevy2)

Jiwoong Youn (youn8)

Peter Meade (meade4)

Michael Blasingame (blasing2)

Zhuoyuan Li (li156)

Qiyue Zhu (zhu72)

Table of Contents

1. Description

With the proliferation of digital pc-game distributors, such as Steam, GOG, and Green Man Gaming, it can be hard to determine which vendor has the best price on a pc-game on any given day. Often times a user may not want to buy a game right away, but be made aware of if/when it goes on sale, and whether or not a sale has historically been the best price on that game. Furthermore, a user may just wish to browse games on sale by genre or reviewer rating, when looking for something new to play on the cheap. Bazaar World of Gaming is a web app which 1) aggregates daily game sales, 2) tracks historical sale data for these games, 3) allows users to manage sale alerts on games, and 4) lets users browse sales on games by genre and reviewer ratings.

2. Process

The process used to develop this project was a variation on XP. We chose XP because we found the general iterative development framework it provided to be helpful and it was familiar to all of us, as most of us had taken CS 427. We however, did make some changes to the bidding process, testing, and collaborative development.

We did not bid on tasks, nor did we hold to a rotating pair scheme. The rationale behind this decision was that different groups of people wanted to work on specific parts of the application, and every sub-group had varying skill-sets. We did not want to place pairs of people who had no real existing skill-synergy together on a part of the project that they would be unfamiliar with or would be incapable,unsuited, and/or unmotivated of completing.

As for testing, we decided to do away with the strict requirement for test-driven development. This is because as a web app that dealt with parsing information from other websites, many components were primarily front-end based, or interacted heavily on a low-level with the back-end databases. As a result, it was unclear how certain functions and features would be defined/implemented until a variety of different approaches were tried and ones that succeeded were identified. Nevertheless, tests were written once implementations were figured out. Naturally, refactoring followed from this.

Finally, while we encouraged collaborative development, we did away with the strict requirement for pair-programming. This was largely a necessity for practical purposes. Some of our group members did not have laptops which made it hard for them to come and program in groups, and certain individuals had significantly restrictive/inflexible schedules due to religious reasons. Allowing these people to collaborate remotely yet implement and program on their own personal schedule helped us get around these issues.

3. Requirements & Specs

A) User Stories:

1. As a user, I want to be able to search for a game by its name
2. As a user, I want to be able to see a game's product details and description
3. As a user, I want to see the metacritic rating of a game if available
4. As a user, I want to be able to browse for games by genre
5. As a user, I want to see what Steam is selling a particular game for
6. As a user, I want to see what Green-Man-Gaming is selling a particular game for
7. As a user, I want to see what Amazon is selling a particular game for
8. As a user, I want to see what Gamers Gate is selling a particular game for
9. As a user, I would like to see sale history on a game
10. As a user, I would like to be able to have an account on the website
11. As a user, I would like to be able to see the cheapest a game is selling for
12. As a user, I would like to be able to browse for games by metacritic rating
13. As a user, I would like to be able to associate my account with a list of games I already own
14. As a user, I would like to be able to associate my account with a price alert for a particular game
15. As a user, I would like to receive an email notifying me when one of my account's price alerts is triggered
16. As a user, I would like to be able to create/edit a user profile

B) Use Cases:

Actor	Task-Level Goal	Priority	More Details
User	Find/Browse/Search for a game	1	link
Database	Respond to game queries	1	link
Parser	Get Sale Data	2	link

Parser	Get Game Product Data	2	
Parser	Get Game Rating from Metacritic	2	link
User	Register/Login for Account	3	link
User	Add Game to Profile	3	
User	Request Sale History for a Game	3	
Website	Show Game Info	1	
Website	Show Game search/browse results	1	
Website	Show Sale Data	3	
Mailer	Send Price Alert	4	
User	Create Price Alert	4	link
Mailer	Email Registration Confirmation	5	

4. Architecture & Design

A) System Overview and Description:

The overall architecture of The Bazaar World of Gaming follows the model–view–controller (MVC) pattern where the front-end views renders user interface and keeps track of user input, models stores application data and controllers send commands between views and models. In detail, the site consists of Game model that stores game information, GameSaleHistory and GameSale model that stores history prices for games and the User model that stores user's information and game collections. Each Game model could have multiple GameSaleHistories from different vendors and dates. Game and User both have corresponding controller that implements searching functionality. Game and User also both have several views that renders corresponding user interfaces that allow users to browse game information and modify their profiles. There's also a PriceAlert model belonging to User which stores price alerts and detects when a price alert is triggered. When a price alert is triggered, it will send the information to the Mailer which will notify the user by email.

Additionally, there are several parsers for game information and sales that will run on the server regularly to add new entries to the database.

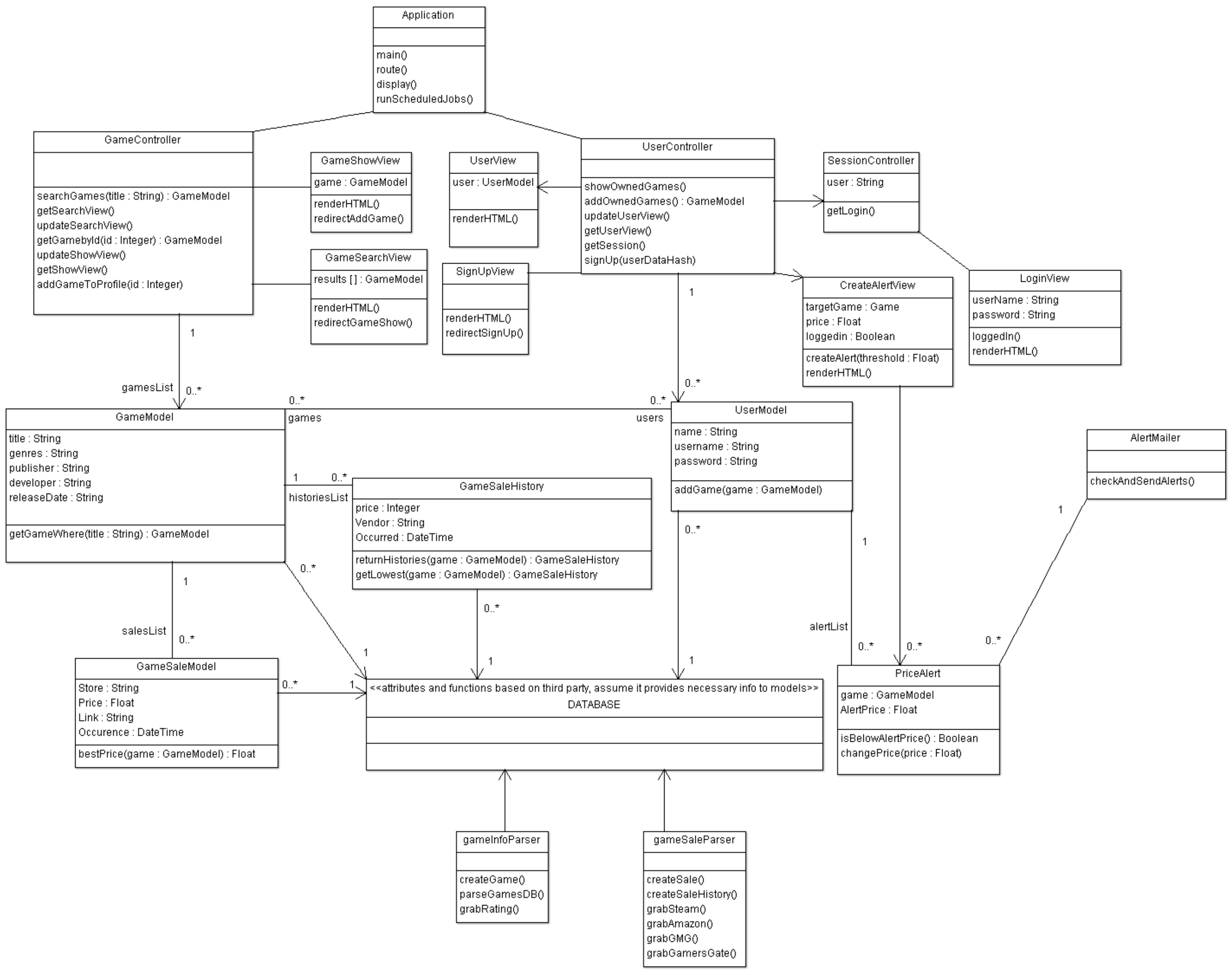

Figure 4.1 System Overview Diagram

B) Sequence Diagrams for Important Functionality:

The core of the site consists of several important sub-systems. Namely user register and login, search and view games and create price alerts. The following sequence diagrams describe the major architecture of each sub-system.

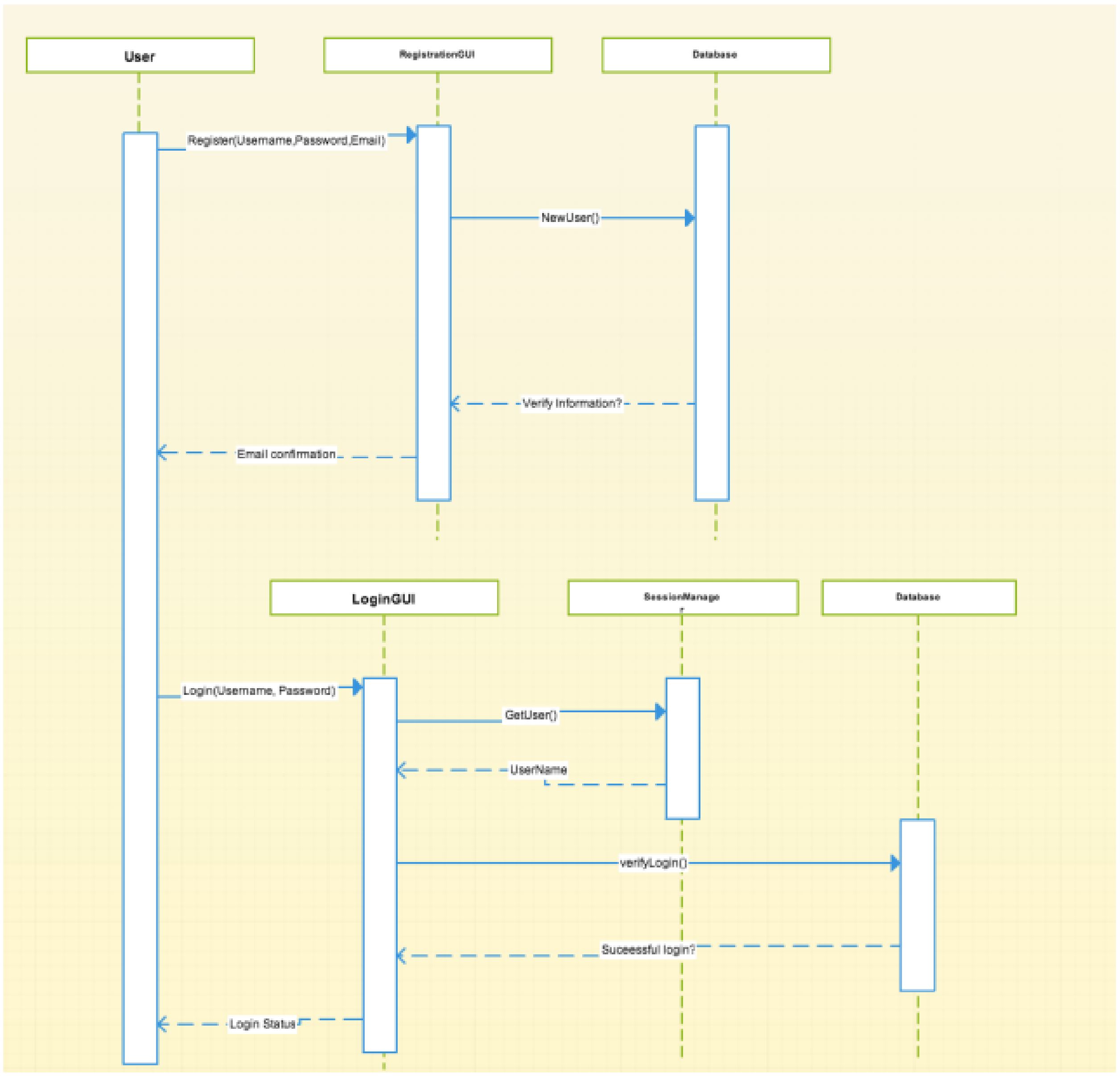

Figure 4.2 How Users Register and Login

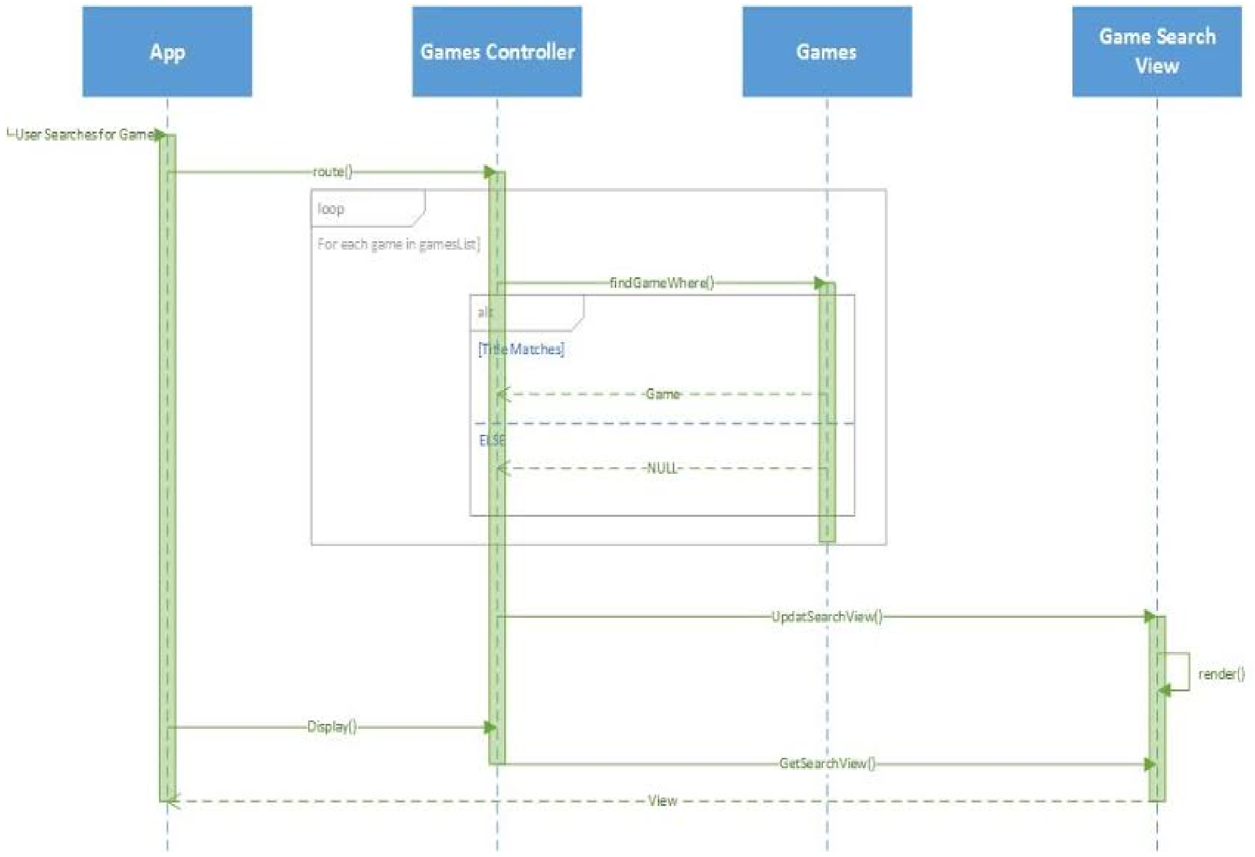

Figure 4.3 How Users Search and View Games and their Sales

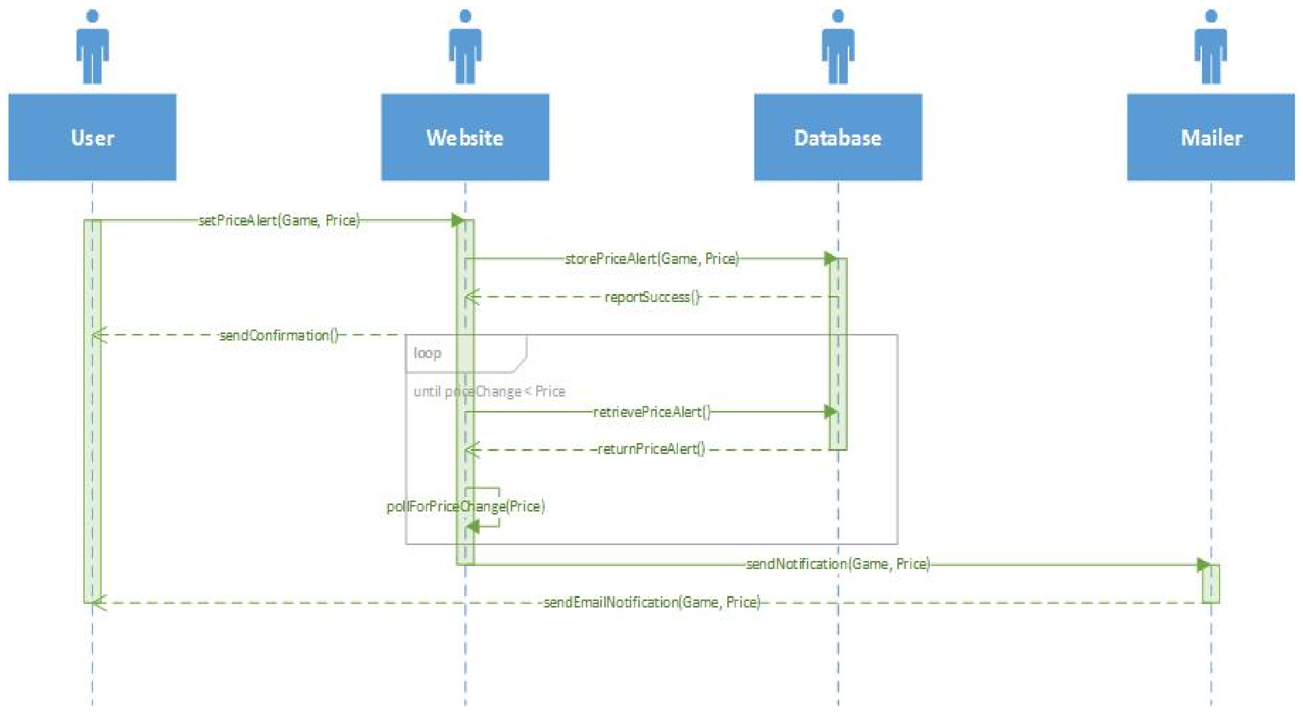
User
Website
Database
Mailer
setPriceAlert(Game, Price)
storePriceAlert(Game, Price)
reportSuccess()
sendConfirmation()
loop
until priceChange < Price
retrievePriceAlert()
returnPriceAlert()
pollForPriceChange(Price)
sendNotification(Game, Price)
sendEmailNotification(Game, Price)

Figure 4.4 How Users Create Price Alerts for Games

C. Backend Models and Helpers:

Models:

1. **`Alert`** - a class that represents an alert and belongs to class Game and User

2. **`Game`** - a class that represents a game and has attributes such as title, description, genres, platform, etc.

3. **`GameSale`** - a class that represents game sale information and has attributes such as occurrence, store, url, original price, and discounted price

4. **`GameSaleHistory`** - a class that represents game sale information that occurred in the past and belongs to class Game

5. **`User`** - a class that represents an user of the website and has attributes such as name, email, username, and password

Helpers:

1. **`AlertHelper`** - a module that manages sending alerts to users

2. **`AmazonHelper`** - a module that parses games off of Amazon's website

3. **`DailyParseHelper`** - a module that contains functions which parses sales from all our vendors, and then runs the alert checker

4. **`GameHelper`** - a module that returns the best price of a game

5. **`GameSearchHelper`** - a module that resolves conflicts in searching for games, as well as naming differences between games that may be the same, but might have different titles according to vendors

6. **`GamersGateHelper`** - a module that parses games off of GamersGate's website

7. **`GamesdbHelper`** - a module that parses games off of the website 'TheGamesDB.net'

8. **`GmgHelper`** - a module that parses games off of Green Man Gaming's website.

9. **`PriceHistoryHelper`** - a module that returns sales histories for a game

10. **`SessionsHelper`** - a module that creates and destroys cookies and controls the current user variable

11. **`SteamHelper`** - a module that parses games off of Steam's website

12. **`StringHelper`** - a module that strips out odd characters in game titles and strings, as well as creates search titles for games, which are used as a primary identifier of games across vendors who might name their games differently

13. **`UsersHelper`** - a module that formats a user's data so that it can easily be displayed on the user's profile page

D. Reasons for Choosing Framework:

We chose to use the Ruby on Rails framework for designing this system, as it allowed for rapid development of a database-backed website. This influenced our design primarily through how RoR makes use of the model, view, controller pattern to represent the entire application. We were able to design our database elements and relations in the form of object oriented classes known as rails models. Thus, all our backend representation and vendor parsers interacted through model interfaces that we designed. This made dividing development between front-end and back-end very efficient, as front-end only had to worry about the structure and interface of the designed back-end models, while back-end only had to worry about creating the models and populating the databases with records of those models. We designed the interaction between front-end user interfaces and the back-end data of the website through controllers in the rails framework, which handled things like routing, registration/signup, searching for games, and creating price alerts. Finally, all front-end of the website was designed through rails views, which consist of html and embedded ruby, rendering content as well passing/receiving handles to back-end models which could then be acted upon through those models' public interfaces.

5. Future Plans & Reflection

Ben Levy:

This project was an excellent learning experience for me, both in the technical side of web-development and use of a framework like RoR and in the area of team development. Since I didn't take 427 before this it was my first exposure to team development with a design methodology. We completed a great deal during the semester but this project can grow in scope commensurate to time and money available. If we had the time (and talent) the GUI could probably be built out better, as it stands it is clearly done by programmers rather than artists. We could expand the sales discovery side of the site, adding additional browsing features. We could also add social features to the site allowing people to share deals that they find interesting. This site seems like it could easily be monetized if it became popular. If we could get a deal with the vendors to get a cut of any sale through our site we would be able to get an income that scales with our user base. Advertising could be placed in an unobstructive way with an option to donate and get rid of the banners.

Arjun Garg:

Through this project, I learned a lot about the Ruby on Rails framework, as well as techniques used behind web-parsing. Additionally, I got much experience with managing and leading a team, especially in the areas of specifying/explaining certain tasks to other people, and working to identify risky areas and ways to mitigate those risks. I think it was overall a good experience, but I think things might've gone smoother had we looked for some kind of iterative process tailored specifically to web development. We felt like a lot of things such as TDD and divided collaborative development were hard to do when making a web-app, because so much of it is fluid, dynamic, and interconnected. Many dependencies on the back-end exist with the front-end, and it isn't exactly easy to test and/or document things like html. Nevertheless, we managed to make it work, but I would be curious if there are certain processes or best practices specifically for iterative web development.

For future plans, the biggest would be to figure out a solid production deployment solution. One of the main challenges with this web application, is that in a production environment, it requires running scheduled jobs to maintain up-to-date pricing data. The routines that would go into these scheduled jobs have of course been written, tested, and demonstrated by our application, and it is not hard to make a cron-job for them. However, most RoR deployment platforms have their own means of creating scheduled jobs, and tend to require financing. This financing could easily be acquired down the road, but we felt it was unsuitable/unnecessary to provide it while we are students of the university, and the professor agreed. As for extra features, we might add a display to the website front end for lowest/popular sales for each vendor. This could help users with serendipitous bargain game purchasing.

Michael Blasingame:

This project has been a great learning experience for me. Not only has it given me a lot of hands-on experience with the Ruby on Rails framework, but also with many web development concepts in general such as MVC and RESTful design. I got to work a lot more on the front end side of things which I haven't really had any experience with. On top of that, it was a good experience with working with a team collaboratively on an original project.

Going forward, I could see this project turning into something that can make money. At this point, I think we've done enough to where our website could be used in real production given that we had the necessary funding to have it deployed on a server that will store all of our database data and run all of the cron-jobs. The front end could use some work seeing as how none of us were really experienced in doing front end development. I also think that some of the features could be cleaned up and enhanced. There's potential to add a social component to the website and I don't think it would be too difficult to implement.

Zach Li:

I learned a lot through this project, and it has been a great experience for me. For me, I have never used Ruby on Rails framework so learning the MVC architecture.was really beneficial for me. I also learned about developing web applications with a big group and using XP programming.Something I think we can improve on in the future is that pair programming while developing web applications seemed tedious and unnecessary. There is no need for two people to work on one aspect of a website. Overall, it was a great learning experience for me and it has been fun working on this project.

In the future, we should improve on the user interface design for the website. The current design of the project is plain and simple; there is no color scheme or stylish design chooses. Since we did not have any front end designers on the team, it was difficult to for us to make the front end look more artistic. In addition, if we had money and time, we can deploy the website after we polish up the UI and backend. It was something we discussed in the meetings that deploying the website will cost money which is a resource that was not provided.

Jiwoong Youn:

Throughout this semester, I learned how to program in Ruby and make a great app using the cool features of Rails. I already have some experience with making a game website supported by a backend database by programming in HTML, CSS, Javascript, and PHP, but I think creating a dynamic website using Ruby on Rails was more impressive and interesting. I also learned that Ruby is a great, dynamic, and developer-friendly language and Rails is even more awesome framework.

I believe all of us did a great job implementing both front-end and back-end part of the application, but I think we could have done better on the overall design of the website. Personally, I love the website so much, but the design is not good enough for being deployed as a real website. Since we had no graphic artists or members who are good at user interface, the current design is the best we can do. If we were more diverse in terms of skill sets, our application may have been more polished and elaborated in design.

Qiyue Zhu:

Through this project, I learned a lot about the Rails framework and web-app developing and, more importantly, how to work with something I've never encountered. I joined the team with almost zero experience in both Ruby on Rails and front-end developing, so I was working on the project while learning the tools and even language I'm using. This experience would certainly encourage me to explore more uncharted territory in web developing or other area.

We accomplished a lot on the project this semester. One thing we could improve is certainly the design of user interface. The current design is simply and neat, but as we don't have any artists in our team, it could certainly be more stylish. And if we are going to take the step to deploy and publish this site, we'll need to put more focus on promoting it. Promoting sometimes could weight more than developing for a project, especially when we are not experts in this area.

Peter Meade:

Much like the rest of my group I enjoyed working on this project for both the experience with web development and the experience with Ruby on Rails. Skill in web development is something that is very useful to have since so many services are offered on the web now. Ruby on Rails being one of the biggest tools for web development means being familiar with it is a great asset as well. Outside of that, this class was much more helpful than its predecessor. Being able to pick your project and process ensures you are interested in what you work on and that the whole development is more engaging.

I think the potential of this project in the future are great. It definitely satisfies a need for the PC gaming community and I could definitely see this project being deployed to the live web. Obviously the UI could benefit from a designers eye but there are other areas that it could be extended as well. I think the database of games can be trimmed a little (currently has games like minesweeper in it). As well I think there are much more relevant features that would improve the quality of the product. An example being integration of some PC gaming services already out that don't directly relate to digital distribution (things like the humble bundle come to mind).